SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

mercuri-protocol

18 Jan 2026

This security assessment report was prepared by SolidityScan.com, a cloud-based Smart Contract Scanner.

Table of Contents

SWEEP TOKEN FUNCTION UNSAFE

UNINITIALIZED OWNERSHIP

UNLIMITED APPROVALS WITHOUT REVOCATION MECHANISM

UNPROTECTED ETHER WITHDRAWAL

ZERO AMOUNT SWAPS NOT REJECTED

BURN ALLOWS BURNING ARBITRARY NFT IF VAULT IS APPROVED

CLOSEPOSITION() USES AMOUNT0MIN/AMOUNT1MIN = 0 DURING LIQUIDITY REMOVAL (NO SLIPPAGE PROTECTION)

CLOSEPOSITION CAN OPERATE ON ARBITRARY TOKENID

CORE DEPENDENCY ADDRESSES ARE NOT CHECKED FOR CONTRACT CODE (MISCONFIGURATION RISK)

DECREASELIQUIDITY ALLOWS OPERATING ON ARBITRARY TOKENID

EVENT NAME TYPO REDUCES OBSERVABILITY AND MONITORING RELIABILITY

MANAGER ADDRESS NOT VALIDATED AGAINST MANAGERREGISTRY AT DEPLOYMENT

MISSING SAFE ERC20 USAGE

MISSPELLED EVENT NAME MANAGERTAPPROVALUPDATED HAMPERS OFF-CHAIN MONITORING AND INTEGRATIONS

POSITION MANAGER–FACTORY MISMATCH NOT VALIDATED

REBALANCEEXACTINPUTSINGLE() LEAVES NON-ZERO ALLOWANCE TO ROUTER AFTER SWAP

REBALANCE CAN ROUTE THROUGH UNINTENDED FEE TIER/POOL

REMOVING LIQUIDITY WITH WEAK MINIMUM CHECKS

SETAPPROVED ALLOWS APPROVING THE ZERO ADDRESS (POTENTIAL DOWNSTREAM MISCONFIGURATION)

SETMANAGER ALLOWS ZERO ADDRESS MANAGER

EVENT BASED REENTRANCY

USE OF FLOATING PRAGMA

LACK OF ZERO VALUE CHECK IN TOKEN TRANSFERS

MISSING EVENTS

MISSING ZERO ADDRESS VALIDATION

NONREENTRANT MODIFIER PLACEMENT

OUTDATED COMPILER VERSION

USE OWNABLE2STEP

ADDING A RETURN STATEMENT WHEN THE FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT

BLOCK VALUES AS A PROXY FOR TIME

CONSTRUCTORS SHOULD EMIT AN EVENT

CONTRACT NAME SHOULD USE PASCALCASE

MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION

MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION

MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS

MISSING INDEXED KEYWORDS IN EVENTS

MISSING @INHERITDOC ON OVERRIDE FUNCTIONS

MISSING NATSPEC COMMENTS IN SCOPE BLOCKS

MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS

MISSING @PARAM IN NATSPEC COMMENTS FOR MODIFIERS

MISSING UNDERSCORE IN NAMING VARIABLES

NAME MAPPING PARAMETERS

VARIABLES SHOULD BE IMMUTABLE

AVOID RE-STORING VALUES

AVOID ZERO-TO-ONE STORAGE WRITES

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

CHEAPER CONDITIONAL OPERATORS

CHEAPER INEQUALITIES IN IF()

DEFAULT INT VALUES ARE MANUALLY RESET

DEFINE CONSTRUCTOR AS PAYABLE

01. Vulnerability Classification and Severity

Description

To enhance navigability, the document is organized in descending order of severity for easy reference. Issues are categorized as ✓ *Fixed*, ⚠ *Pending Fix*, or *Won't Fix*, indicating their current status. *Won't Fix* denotes that the team is aware of the issue but has chosen not to resolve it. Issues labeled as ⚠ *Pending Fix* state that the bug is yet to be resolved. Additionally, each issue's severity is assessed based on the risk of exploitation or the potential for other unexpected or unsafe behavior.

● Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

● High

High-severity vulnerabilities pose a significant risk to both the Smart Contract and the organization. They can lead to user fund losses, may have conditional requirements, and are challenging to exploit.

● Medium

The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.

● Low

The issue has minimal impact on the contract's ability to operate.

● Informational

The issue does not affect the contract's operational capability but is considered good practice to address.

● Gas

This category deals with optimizing code and refactoring to conserve gas.

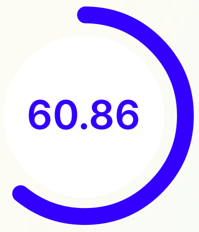
02. Executive Summary

mercuri-protocol

Github Project

https://github.com/mercuri-finance/mercuri-protocol

Language	Audit Methodology	Commit Hash
Solidity	**Static Scanning**	-
Website	Publishers/Owner Name	Organization
-	-	-
Contact Email		
-		



Security Score is AVERAGE

The SolidityScan score is calculated based on lines of code and weights assigned to each issue depending on the severity and confidence. To improve your score, view the detailed result and leverage the remediation solutions provided.

This report has been prepared for mercuri-protocol using SolidityScan to scan and discover vulnerabilities and safe coding practices in their smart contract including the libraries used by the contract that are not officially recognized. The SolidityScan tool runs a comprehensive static analysis on the Solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over 700+ modules. The scanning and auditing process covers the following areas:

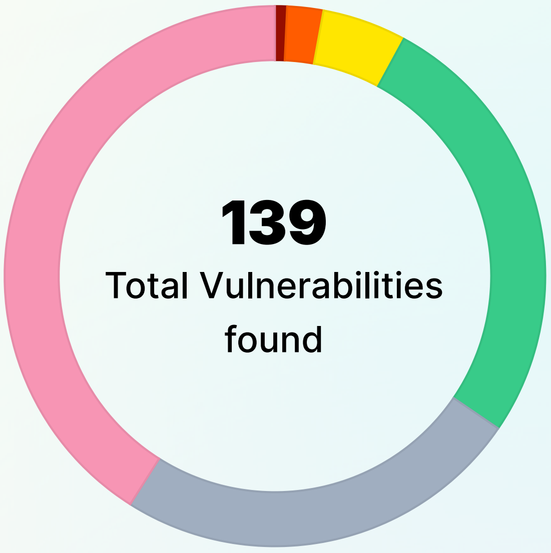
Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date.

The SolidityScan Team recommends running regular audit scans to identify any vulnerabilities that are introduced after mercuri-protocol introduces new features or refactors the code.

03. Findings Summary

mercuri-protocol
View on Github

Security Score	Scan duration	Lines of code
60.86/100	702 secs	635



1	3	7	37	34	57
Crit	High	Med	Low	Info	Gas

This audit report has not been verified by the SolidityScan team. To learn more about our published reports. click here

ACTION TAKEN

91	0	0	142
✓ *Fixed*	*False Positive*	*Won't Fix*	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
C001	● Critical	COLLECT() ALLOWS ARBITRARY RECIPIENT, ENABLING MANAGER TO DRAIN VAULT ASSETS (FEES AND PRINCIPAL)	1	SolidityScan AI	⚠ *Pending Fix*
C002	● Critical	MANAGER CAN DRAIN VAULT ASSETS VIA ARBITRARY RECIPIENT IN COLLECT()	1	SolidityScan AI	✓ *Fixed*
H001	● High	INSUFFICIENT POOL AUTHENTICITY VERIFICATION ALLOWS MALICIOUS NON-UNISWAP POOLS	1	SolidityScan AI	✓ *Fixed*
H002	● High	MINT CAN EXFILTRATE FUNDS BY MINTING NFT TO ARBITRARY RECIPIENT	1	SolidityScan AI	✓ *Fixed*
H003	● High	PERFORMANCE FEE INCORRECTLY APPLIED TO PRINCIPAL DUE TO CALL ORDER IN CLOSEPOSITION	1	SolidityScan AI	✓ *Fixed*
H004	● High	POOL FACTORY PARAMETER VALIDATION WEAK	1	SolidityScan AI	⚠ *Pending Fix*
H005	● High	PROTOCOL PERFORMANCE FEE BYPASS VIA PRE-COLLECTING OWED FEES	1	SolidityScan AI	⚠ *Pending Fix*
H006	● High	REENTRANCY	1	Automated	⚠ *Pending Fix*
H007	● High	WITHDRAWAL QUEUE ORDERING BUGS	1	SolidityScan AI	✓ *Fixed*
M001	● Medium	ETH WETH UNWRAP LOGIC FRAGILE	1	SolidityScan AI	✓ *Fixed*
M002	● Medium	FEE MECHANISM VULNERABILITIES	1	SolidityScan AI	✓ *Fixed*
M003	● Medium	INCREASELIQUIDITY LACKS TOKENID SCOPING ALLOWING VALUE LEAKAGE TO THIRD-PARTY POSITIONS	1	SolidityScan AI	✓ *Fixed*
M004	● Medium	MISSING ZERO-ADDRESS VALIDATION FOR SWAP_ROUTER IN CONSTRUCTOR	1	SolidityScan AI	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
M005	● Medium	APPROVE FRONT-RUNNING ATTACK	1	Automated	✓ *Fixed*
M006	● Medium	DEPRECATED SAFEAPPROVE	2	Automated	! *Partially fixed*
M007	● Medium	SWAP FEE UPPER BOUND NOT ENFORCED	2	SolidityScan AI	✓ *Fixed*
M008	● Medium	SWEEP TOKEN FUNCTION UNSAFE	1	SolidityScan AI	⚠ *Pending Fix*
M009	● Medium	UNINITIALIZED OWNERSHIP	2	SolidityScan AI	! *Partially fixed*
M010	● Medium	UNLIMITED APPROVALS WITHOUT REVOCATION MECHANISM	1	SolidityScan AI	✓ *Fixed*
M011	● Medium	UNPROTECTED ETHER WITHDRAWAL	2	SolidityScan AI	✓ *Fixed*
M012	● Medium	ZERO AMOUNT SWAPS NOT REJECTED	1	SolidityScan AI	✓ *Fixed*
L001	● Low	BURN ALLOWS BURNING ARBITRARY NFT IF VAULT IS APPROVED	1	SolidityScan AI	✓ *Fixed*
L002	● Low	CLOSEPOSITION() USES AMOUNT0MIN/AMOUNT1MIN = 0 DURING LIQUIDITY REMOVAL (NO SLIPPAGE PROTECTION)	1	SolidityScan AI	⚠ *Pending Fix*
L003	● Low	CLOSEPOSITION CAN OPERATE ON ARBITRARY TOKENID	1	SolidityScan AI	✓ *Fixed*
L004	● Low	CORE DEPENDENCY ADDRESSES ARE NOT CHECKED FOR CONTRACT CODE (MISCONFIGURATION RISK)	1	SolidityScan AI	✓ *Fixed*
L005	● Low	DECREASELIQUIDITY ALLOWS OPERATING ON ARBITRARY TOKENID	1	SolidityScan AI	✓ *Fixed*
L006	● Low	EVENT NAME TYPO REDUCES OBSERVABILITY AND MONITORING RELIABILITY	1	SolidityScan AI	✓ *Fixed*
L007	● Low	MANAGER ADDRESS NOT VALIDATED AGAINST MANAGERREGISTRY AT DEPLOYMENT	1	SolidityScan AI	✓ *Fixed*
L008	● Low	MISSING SAFE ERC20 USAGE	1	SolidityScan AI	✓ *Fixed*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
L009	● Low	MISSPELLED EVENT NAME MANAGERTAPPROVALUPDATED HAMPERS OFF-CHAIN MONITORING AND INTEGRATIONS	1	SolidityScan AI	⚠ *Pending Fix*
L010	● Low	POSITION MANAGER–FACTORY MISMATCH NOT VALIDATED	1	SolidityScan AI	✓ *Fixed*
L011	● Low	REBALANCEEXACTINPUTSINGLE() LEAVES NON-ZERO ALLOWANCE TO ROUTER AFTER SWAP	1	SolidityScan AI	⚠ *Pending Fix*
L012	● Low	REBALANCE CAN ROUTE THROUGH UNINTENDED FEE TIER/POOL	1	SolidityScan AI	✓ *Fixed*
L013	● Low	REMOVING LIQUIDITY WITH WEAK MINIMUM CHECKS	3	SolidityScan AI	✓ *Fixed*
L014	● Low	SETAPPROVED ALLOWS APPROVING THE ZERO ADDRESS (POTENTIAL DOWNSTREAM MISCONFIGURATION)	1	SolidityScan AI	✓ *Fixed*
L015	● Low	SETMANAGER ALLOWS ZERO ADDRESS MANAGER	1	SolidityScan AI	✓ *Fixed*
L016	● Low	EVENT BASED REENTRANCY	1	Automated	⚠ *Pending Fix*
L017	● Low	USE OF FLOATING PRAGMA	9	Automated	! *Partially fixed*
L018	● Low	LACK OF ZERO VALUE CHECK IN TOKEN TRANSFERS	5	Automated	! *Partially fixed*
L019	● Low	MISSING EVENTS	9	Automated	! *Partially fixed*
L020	● Low	MISSING ZERO ADDRESS VALIDATION	5	Automated	! *Partially fixed*
L021	● Low	NONREENTRANT MODIFIER PLACEMENT	9	Automated	⚠ *Pending Fix*
L022	● Low	OUTDATED COMPILER VERSION	9	Automated	! *Partially fixed*
L023	● Low	USE OWNABLE2STEP	1	Automated	⚠ *Pending Fix*
I001	● Informational	ADDING A RETURN STATEMENT WHEN THE FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT	3	Automated	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I002	● Informational	BLOCK VALUES AS A PROXY FOR TIME	2	Automated	*Pending Fix*
I003	● Informational	CONSTRUCTORS SHOULD EMIT AN EVENT	1	Automated	*Pending Fix*
I004	● Informational	CONTRACT NAME SHOULD USE PASCALCASE	1	Automated	✓ *Fixed*
I005	● Informational	MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	9	Automated	*Partially fixed*
I006	● Informational	MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	4	Automated	*Partially fixed*
I007	● Informational	MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS	15	Automated	*Partially fixed*
I008	● Informational	MISSING INDEXED KEYWORDS IN EVENTS	1	Automated	*Pending Fix*
I009	● Informational	MISSING @INHERITDOC ON OVERRIDE FUNCTIONS	3	Automated	✓ *Fixed*
I010	● Informational	MISSING NATSPEC COMMENTS IN SCOPE BLOCKS	14	Automated	*Pending Fix*
I011	● Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS	3	Automated	✓ *Fixed*
I012	● Informational	MISSING @PARAM IN NATSPEC COMMENTS FOR MODIFIERS	3	Automated	*Pending Fix*
I013	● Informational	MISSING UNDERSCORE IN NAMING VARIABLES	4	Automated	*Partially fixed*
I014	● Informational	NAME MAPPING PARAMETERS	3	Automated	*Pending Fix*
I015	● Informational	VARIABLES SHOULD BE IMMUTABLE	1	Automated	*Pending Fix*
G001	● Gas	AVOID RE-STORING VALUES	2	Automated	*Pending Fix*
G002	● Gas	AVOID ZERO-TO-ONE STORAGE WRITES	5	Automated	*Pending Fix*
G003	● Gas	CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE	7	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
G004	● Gas	CHEAPER CONDITIONAL OPERATORS	10	Automated	⚠ *Pending Fix*
G005	● Gas	CHEAPER INEQUALITIES IN IF()	9	Automated	⚠ *Pending Fix*
G006	● Gas	DEFAULT INT VALUES ARE MANUALLY RESET	3	Automated	⚠ *Pending Fix*
G007	● Gas	DEFINE CONSTRUCTOR AS PAYABLE	3	Automated	⚠ *Pending Fix*
G008	● Gas	REVERTING FUNCTIONS CAN BE PAYABLE	5	Automated	⚠ *Pending Fix*
G009	● Gas	FUNCTION SHOULD RETURN STRUCT	3	Automated	! *Partially fixed*
G010	● Gas	GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION	3	Automated	⚠ *Pending Fix*
G011	● Gas	SIMILAR DATATYPES CAN BE PACKED TOGETHER	3	Automated	✓ *Fixed*
G012	● Gas	SMALLER DATA TYPES COST MORE	2	Automated	⚠ *Pending Fix*
G013	● Gas	SPLITTING REQUIRE STATEMENTS	2	Automated	⚠ *Pending Fix*
G014	● Gas	STORAGE VARIABLE CACHING IN MEMORY	21	Automated	⚠ *Pending Fix*
G015	● Gas	UNUSED IMPORTS	8	Automated	! *Partially fixed*
G016	● Gas	VARIABLES DECLARED BUT NEVER USED	1	Automated	✓ *Fixed*

04. Vulnerability Details

Issue Type

COLLECT() ALLOWS ARBITRARY RECIPIENT, ENABLING MANAGER TO DRAIN VAULT ASSETS (FEES AND PRINCIPAL)

S. No.	Severity	Detection Method	Instances
C001	● Critical	SolidityScan AI	1

Bug ID	File Location	Line No.	Action Taken
SSP_121239_222	--	--	*Pending Fix*

Upgrade your Plan to view the full report

1 Critical Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

ADDING A RETURN STATEMENT WHEN THE FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT

S. No.	Severity	Detection Method	Instances
I001	● Informational	Automated	3

Bug ID	File Location	Line No.	Action Taken
SSP_121239_1	--	--	⚠ *Pending Fix*
SSP_121239_2	--	--	⚠ *Pending Fix*
SSP_121239_3	--	--	⚠ *Pending Fix*

Upgrade your Plan to view the full report

3 Informational Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G001	● Gas	Automated	2

Description

The function is found to be allowing re-storing the value in the contract's state variable even when the old value is eq ual to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operation (2900 ga s), which could be avoided. If the old value and the new value are the same, not updating the storage would avoid thi s cost and could instead incur a `Gcoldsload` (2100 gas) or a `Gwarmaccess` (100 gas), potentially saving gas.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_22	contracts/Vault.sol	L166 - L169	⚠ *Pending Fix*
SSP_121239_23	contracts/interf...gistry.sol	L38 - L41	⚠ *Pending Fix*

Issue Type

AVOID ZERO-TO-ONE STORAGE WRITES

S. No.	Severity	Detection Method	Instances
G002	● Gas	Automated	5

Description

Writing a storage variable from zero to a non-zero value costs 22,100 gas (20,000 for the write and 2,100 for cold acc ess), making it one of the most expensive operations. This is why patterns like OpenZeppelin's `ReentrancyGuard` us e `1` and `2` instead of `0` and `1`—to avoid the high cost of zero-to-non-zero writes. Non-zero to non-zero updates cost only 5,000 gas.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_6	contracts/Vault.sol	L157 - L157	*Pending Fix*
SSP_121239_7	contracts/Vault.sol	L282 - L282	*Pending Fix*
SSP_121239_8	contracts/Vault.sol	L353 - L353	*Pending Fix*
SSP_121239_9	contracts/Vault.sol	L429 - L429	*Pending Fix*
SSP_121239_9	contracts/Vault.sol	L473 - L473	*Pending Fix*

Issue Type

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

S. No.	Severity	Detection Method	Instances
G003	● Gas	Automated	7

Description

The repeated usage of `address(this)` within the contract could result in increased gas costs due to multiple executions of the same computation, potentially impacting efficiency and overall transaction expenses.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_42	contracts/Vault.sol	L188 - L188	⚠ *Pending Fix*
SSP_121239_43	contracts/Vault.sol	L195 - L195	⚠ *Pending Fix*
SSP_121239_44	contracts/Vault.sol	L228 - L228	⚠ *Pending Fix*
SSP_121239_44	contracts/Vault.sol	L464 - L464	⚠ *Pending Fix*
SSP_121239_45	contracts/Vault.sol	L275 - L275	⚠ *Pending Fix*
SSP_121239_46	contracts/Vault.sol	L305 - L305	⚠ *Pending Fix*
SSP_121239_47	contracts/Vault.sol	L490 - L490	⚠ *Pending Fix*

Issue Type

CHEAPER CONDITIONAL OPERATORS

S. No.	Severity	Detection Method	Instances
G004	● Gas	Automated	10

Description

During compilation, `x != 0` is cheaper than `x > 0` for unsigned integers in solidity inside conditional statements.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_128	contracts/Vault.sol	L184 - L184	⚠ *Pending Fix*
SSP_121239_129	contracts/Vault.sol	L191 - L191	⚠ *Pending Fix*
SSP_121239_130	contracts/Vault.sol	L237 - L237	⚠ *Pending Fix*
SSP_121239_131	contracts/Vault.sol	L238 - L238	⚠ *Pending Fix*
SSP_121239_132	contracts/Vault.sol	L258 - L258	⚠ *Pending Fix*
SSP_121239_133	contracts/Vault.sol	L341 - L341	⚠ *Pending Fix*
SSP_121239_133	contracts/Vault.sol	L372 - L372	⚠ *Pending Fix*
SSP_121239_134	contracts/Vault.sol	L345 - L345	⚠ *Pending Fix*
SSP_121239_134	contracts/Vault.sol	L376 - L376	⚠ *Pending Fix*
SSP_121239_135	contracts/Vault.sol	L447 - L447	⚠ *Pending Fix*

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G005	● Gas	Automated	9

Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Bug ID	File Location	Line No.	Action Taken
SSP_121239_71	contracts/Vault.sol	L184 - L184	*Pending Fix*
SSP_121239_72	contracts/Vault.sol	L191 - L191	*Pending Fix*
SSP_121239_73	contracts/Vault.sol	L237 - L237	*Pending Fix*
SSP_121239_74	contracts/Vault.sol	L238 - L238	*Pending Fix*
SSP_121239_75	contracts/Vault.sol	L341 - L341	*Pending Fix*
SSP_121239_75	contracts/Vault.sol	L372 - L372	*Pending Fix*
SSP_121239_76	contracts/Vault.sol	L345 - L345	*Pending Fix*
SSP_121239_76	contracts/Vault.sol	L376 - L376	*Pending Fix*
SSP_121239_77	contracts/Vault.sol	L447 - L447	*Pending Fix*

Issue Type

DEFAULT INT VALUES ARE MANUALLY RESET

S. No.	Severity	Detection Method	Instances
G006	● Gas	Automated	3

Description

The contract is found to inefficiently reset integer variables to their default value of zero using manual assignment. In Solidity, manually setting a variable to its default value does not free up storage space, leading to unnecessary gas c onsumption. Instead, using the `.delete` keyword can achieve the same result while also freeing up storage space on the Ethereum blockchain, resulting in gas cost savings.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_24	contracts/Vault.sol	L282 - L282	⚠ *Pending Fix*
SSP_121239_25	contracts/Vault.sol	L429 - L429	⚠ *Pending Fix*
SSP_121239_25	contracts/Vault.sol	L473 - L473	⚠ *Pending Fix*

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G007	● Gas	Automated	3

Description

Developers can save around 10 opcodes and some gas if the constructors are defined as payable.
However, it should be noted that it comes with risks because payable constructors can accept ETH during deployme nt.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_39	contracts/VaultFactory.sol	L64 - L86	*Pending Fix*
SSP_121239_40	contracts/Vault.sol	L127 - L158	*Pending Fix*
SSP_121239_41	contracts/interf...gistry.sol	L30 - L32	*Pending Fix*

Issue Type

REVERTING FUNCTIONS CAN BE PAYABLE

S. No.	Severity	Detection Method	Instances
G008	● Gas	Automated	5

Description

If a function modifier such as `onlyOwner` is used, the function will revert if a normal user tries to pay the function. Marking the function as payable will lower the gas cost for legitimate callers because the compiler will not include checks for whether a payment was provided.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_84	contracts/VaultFactory.sol	L92 - L104	*Pending Fix*
SSP_121239_85	contracts/Vault.sol	L166 - L169	*Pending Fix*
SSP_121239_86	contracts/Vault.sol	L251 - L287	*Pending Fix*
SSP_121239_87	contracts/Vault.sol	L294 - L297	*Pending Fix*
SSP_121239_88	contracts/interf...gistry.sol	L38 - L41	*Pending Fix*

Issue Type

FUNCTION SHOULD RETURN STRUCT

S. No.	Severity	Detection Method	Instances
G009	● Gas	Automated	1

Description

The function was detected to be returning multiple values.
Consider using a `struct` instead of multiple return values for the function. It can improve code readability.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_68	contracts/Vault.sol	L329 - L354	*Pending Fix*

Issue Type

GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION

S. No.	Severity	Detection Method	Instances
G010	● Gas	Automated	3

Description

The contract is found to use multiple operands within a single `if` or `else if` statement, which can lead to unnecessar y gas consumption due to the way the EVM evaluates compound boolean expressions. Each operand in a compound condition is evaluated even if the first condition fails, unless short-circuiting occurs, and the combined logic can resul t in more complex bytecode and higher gas usage compared to using nested `if` statements. This inefficiency is parti cularly relevant in functions that are called frequently or within loops.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_151	contracts/Vault.sol	L184 - L189	⚠ *Pending Fix*
SSP_121239_152	contracts/Vault.sol	L191 - L196	⚠ *Pending Fix*
SSP_121239_153	contracts/Vault.sol	L223 - L223	⚠ *Pending Fix*

Issue Type

SMALLER DATA TYPES COST MORE

S. No.	Severity	Detection Method	Instances
G012	● Gas	Automated	2

Description

Usage of smaller integer types such as `uint8`, `uint16`, `int8`, or `int16` in arithmetic operations incur additional gas costs compared to the default `uint` and `int` types, which are typically `uint256` and `int256` respectively.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_125	contracts/Vault.sol	L234 - L234	*Pending Fix*
SSP_121239_126	contracts/Vault.sol	L235 - L235	*Pending Fix*

Issue Type

SPLITTING REQUIRE STATEMENTS

S. No.	Severity	Detection Method	Instances
G013	● Gas	Automated	2

Description

Require statements when combined using operators in a single statement usually lead to a larger deployment gas co st but with each runtime calls, the whole thing ends up being cheaper by some gas units.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_172	contracts/Vault.sol	L337 - L337	*Pending Fix*
SSP_121239_173	contracts/Vault.sol	L492 - L496	*Pending Fix*

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G014	● Gas	Automated	21

Description

The contract is using the state variable multiple times in the function.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Bug ID	File Location	Line No.	Action Taken
SSP_121239_174	contracts/Vault.sol	L179 - L200	*Pending Fix*
SSP_121239_174	contracts/Vault.sol	L179 - L200	*Pending Fix*
SSP_121239_174	contracts/Vault.sol	L179 - L200	*Pending Fix*
SSP_121239_175	contracts/Vault.sol	L209 - L241	*Pending Fix*
SSP_121239_176	contracts/Vault.sol	L251 - L287	*Pending Fix*
SSP_121239_176	contracts/Vault.sol	L251 - L287	*Pending Fix*
SSP_121239_177	contracts/Vault.sol	L304 - L317	*Pending Fix*
SSP_121239_177	contracts/Vault.sol	L304 - L317	*Pending Fix*
SSP_121239_178	contracts/Vault.sol	L329 - L354	*Pending Fix*
SSP_121239_178	contracts/Vault.sol	L329 - L354	*Pending Fix*
SSP_121239_178	contracts/Vault.sol	L329 - L354	*Pending Fix*

Bug ID	File Location	Line No.	Action Taken
SSP_121239_179	contracts/Vault.sol	L364 - L382	⚠ *Pending Fix*
SSP_121239_179	contracts/Vault.sol	L364 - L382	⚠ *Pending Fix*
SSP_121239_179	contracts/Vault.sol	L364 - L382	⚠ *Pending Fix*
SSP_121239_180	contracts/Vault.sol	L423 - L430	⚠ *Pending Fix*
SSP_121239_181	contracts/Vault.sol	L439 - L474	⚠ *Pending Fix*
SSP_121239_181	contracts/Vault.sol	L439 - L474	⚠ *Pending Fix*
SSP_121239_182	contracts/Vault.sol	L482 - L502	⚠ *Pending Fix*
SSP_121239_182	contracts/Vault.sol	L482 - L502	⚠ *Pending Fix*
SSP_121239_182	contracts/Vault.sol	L482 - L502	⚠ *Pending Fix*
SSP_121239_183	contracts/Vault.sol	L508 - L519	⚠ *Pending Fix*

Issue Type

UNUSED IMPORTS

S. No.	Severity	Detection Method	Instances
G015	● Gas	Automated	2

Description

Solidity is a Gas-constrained language. Having unused code or import statements incurs extra gas usage when deplo ying the contract.

Bug ID	File Location	Line No.	Action Taken
SSP_121239_156	contracts/VaultFactory.sol	L4 - L4	⚠ *Pending Fix*
SSP_121239_157	contracts/VaultFactory.sol	L5 - L5	⚠ *Pending Fix*

05. Scan History

● Critical ● High ● Medium ● Low ● Informational ● Gas

No	Date	Security Score	Scan Overview
1.	2026-01-18	**60.86**	● 1 ● 3 ● 7 ● 37 ● 34 ● 57
2.	2026-01-18	**60.77**	● 1 ● 5 ● 11 ● 59 ● 65 ● 69
3.	2026-01-18	**72.03**	● 0 ● 1 ● 3 ● 46 ● 65 ● 69

06. Disclaimer

The Reports neither endorse nor condemn any specific project or team, nor do they guarantee the security of any specific project. The contents of this report do not, and should not be interpreted as having any bearing on, the economics of tokens, token sales, or any other goods, services, or assets.

The security audit is not meant to replace functional testing done before a software release.

There is no warranty that all possible security issues of a particular smart contract(s) will be found by the tool, i.e., It is not guaranteed that there will not be any further findings based solely on the results of this evaluation.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. There is no warranty or representation made by this report to any Third Party in regards to the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business.

In no way should a third party use these reports to make any decisions about buying or selling a token, product, service, or any other asset. It should be noted that this report is not investment advice, is not intended to be relied on as investment advice, and has no endorsement of this project or team. It does not serve as a guarantee as to the project's absolute security.

The assessment provided by SolidityScan is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. SolidityScan owes no duty to any third party by virtue of publishing these Reports.

As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits including manual audit and a public bug bounty program to ensure the security of the smart contracts.